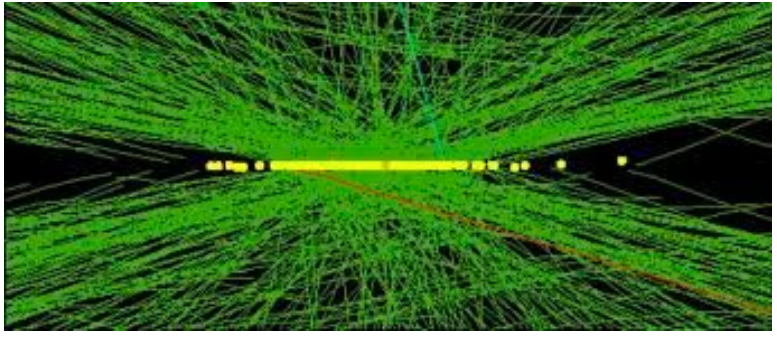
GNN Tracking

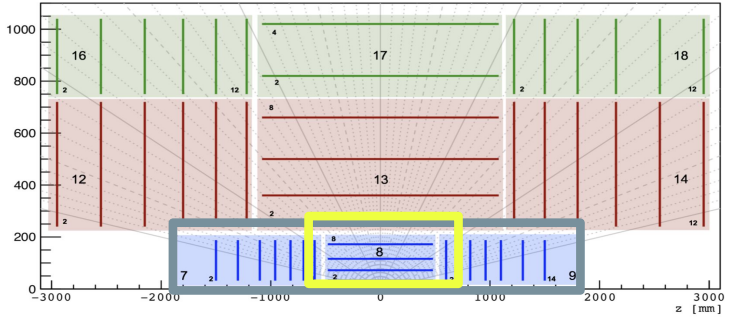
Markus Atkinson, Gage DeZoort, Lindsey Gray, Mark Neubauer, Isobel Ojalvo, Savannah Thais

IRIS HEP Retreat 05/28/2020

Project Introduction

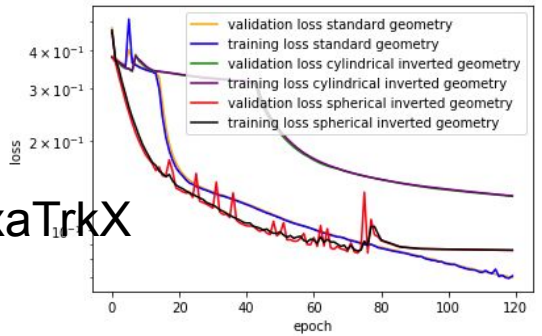
- Our group is aiming to improve charged-particle tracking in ATLAS & CMS → one of the primary challenges for the HL-LHC era
  - Particularly through the use of accelerators (FPGAs) and ML algorithms such as Graph Neural Networks (GNNs)
- Relatively new project (~6 months old)
- Collaboration between Princeton and Illinois
  - Lindsey Gray (FNAL) collaborates through our meetings, Slack discussions, sharing of code & ideas
- Have bi-weekly meetings among the team
  - Developing internal GitHub with visualizations (to contribute to discussed repo)
- Beginning to contribute the ExaTrkX Project (e.g. developing shared repo based on MLFlow, with Ben G. helping out) and attend their meetings, report on our progress
- Currently focused only on ‘pixel detector’ using the TrackML dataset
  - Both ATLAS and CMS use pixel trackers for track seeding and vertex reconstruction
  - Stubs or ‘tracklets’ from the pixel detector are used to seed first pass inside-out track finding

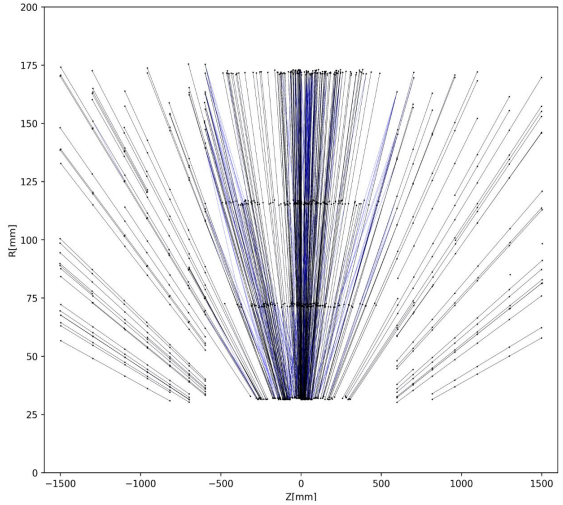


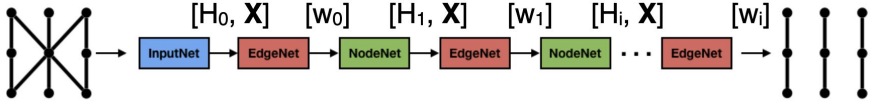


Achievements So Far

- Developed visualization tools for graphs
- Using a layer pair approach for graph construction from ExaTrkX
  - Developing and optimizing cuts in 2 existing graph constructions
  - Measured efficiencies of implemented graph constructions
  - Explored two specific transformed geometries
  - Including different modules in the graphs (barrel, endcaps)
- Investigated multiple GNN architectures for training
  - Using a recurrent combination of edge and node classifiers
  - Using integrated graph modules
  - Implemented Interaction Network GNN (in early stage of study)
- Implemented initial track building algorithms
  - Using Union Find
  - Preliminary results for tracking efficiencies and fake rates
- Acquire/access necessary software/hardware for FPGA development
  - Vivado (Xilinx), Quartus (Intel), HLS, OpenCL
  - Xilinx Alveo Card, Intel Stratix 10

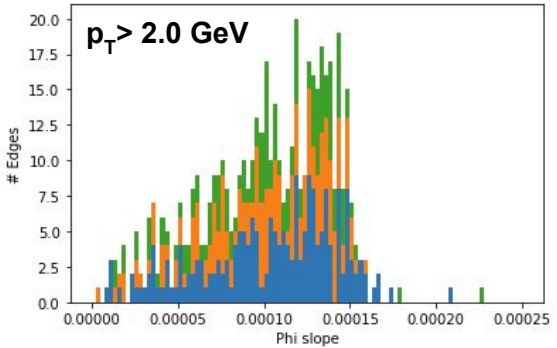


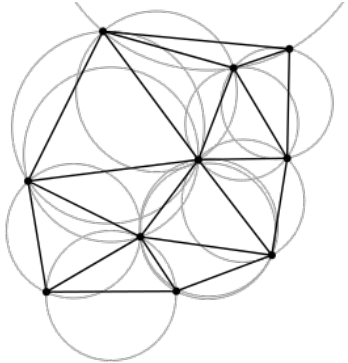


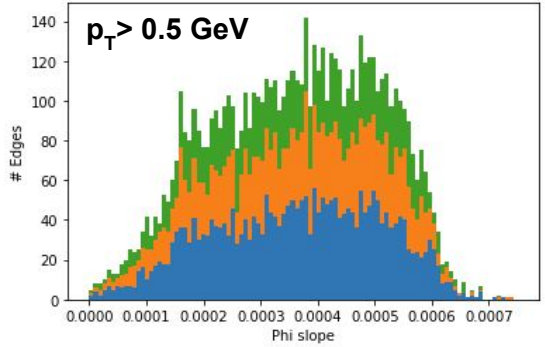


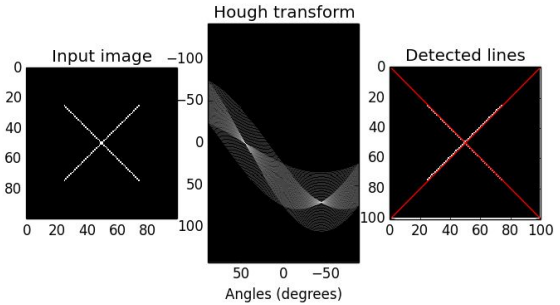
On Going & Planned Work

- Optimize edge cutting parameters
  - $\Delta\phi/\Delta R$, k of kNN
- Implement improved track-finding algorithms:
  - DBScan
  - Integrate into GNN architecture
- Explore additional graph construction algorithms:
  - Dynamic knn graphs
  - Simulated annealing
  - Modified triangulation
  - Dynamic point clouds
- Explore additional architectures:
  - GraphSAGE and PinSAGE
  - Spectral Convolutions
- Explore other transforms and embeddings
  - Hough Transform
- Segment Detector (if necessary)
  - Reduce graph size, requires additional post-processing
- Explore how these techniques and code developed on trackML data generalize to **ATLAS/CMS** simulation









FPGA Acceleration

- Implement traditional (non-ML based) tracking algorithm on FPGA
- Define GNN algorithm components to be implemented
- Some promising work on fitting GNNs into FPGA resource constraints by the HLS4ML group (e.g. see CTD talk by Y. Iiyama using the GarNet model)
- Comparisons across architectures
  - GPUs vs FPGAs
- Princeton group is using OpenCL and an Intel Stratis 10 Dev Kit
- Illinois group is using HLS and Xilinx Alveo U250
  - Collaboration with NCSA Innovative System Laboratory

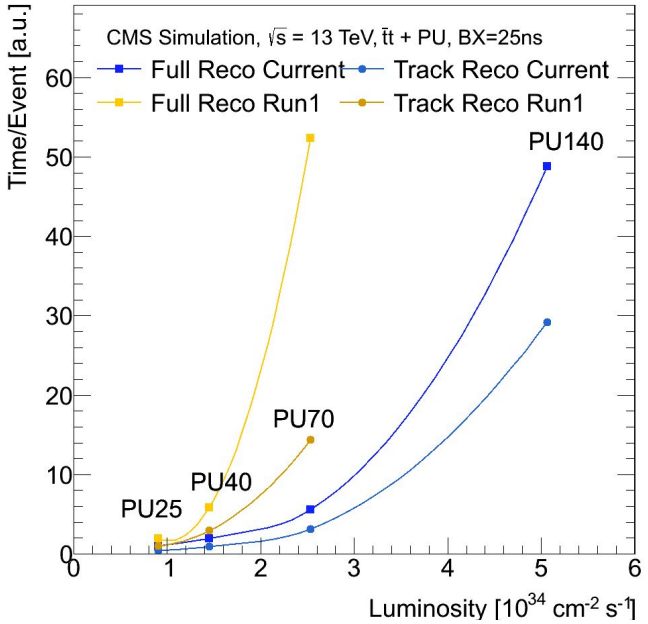
Year 3 and 4/5 Milestones (proposed, for discussion!)

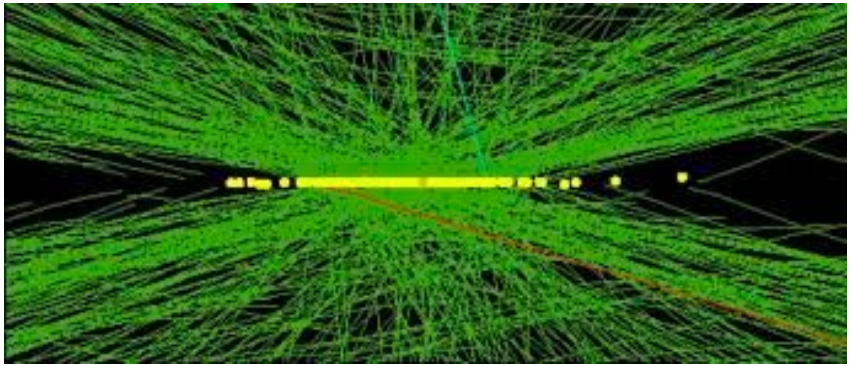
- Year 3 Milestones & Deliverables
  - 6-9 months: Explore implementation of traditional (non-ML) tracking algorithms on FPGAs
  - 9-12 months: Converge on an efficient and ‘acceleratable’ graph-based tracking pipeline
    - Graph design (embeddings, transformations, edge constructions)
    - GNN architecture (ML method)
    - Track construction/post-processing
  - 12-18 months: Snowmass white paper
- Year 4/5 goals
  - Y4 Demonstration and benchmarking of our graph-based tracking accelerator in an HLT application using SSL resources
  - Y5 Our project integrated into the planning for one or both of the experiments
- Metrics
  - These need to be developed but could include e.g. the level of event tracking time reduction (e.g. fast seeding using the pixel detector), broader impacts to non-LHC-trigger applications, etc.

Backup

Tracking Challenge at HL-LHC

- HL-LHC poses increased challenges to tracking
  - 140-200 pileup, ~10,000 tracks every 25 ns
  - 10^{35} $cm^{-2}s^{-1}$ instantaneous luminosity
- Tracking is the most computationally intensive reco task
  - Time grows exponentially with increasing pile up
  - Additional challenges of overlapping tracks
- Must exploit developments in hardware and software
  - Improved algorithms and data representation
  - Parallelize currently serial algorithms
  - Adapt to modern architectures (GPU, FPGA)



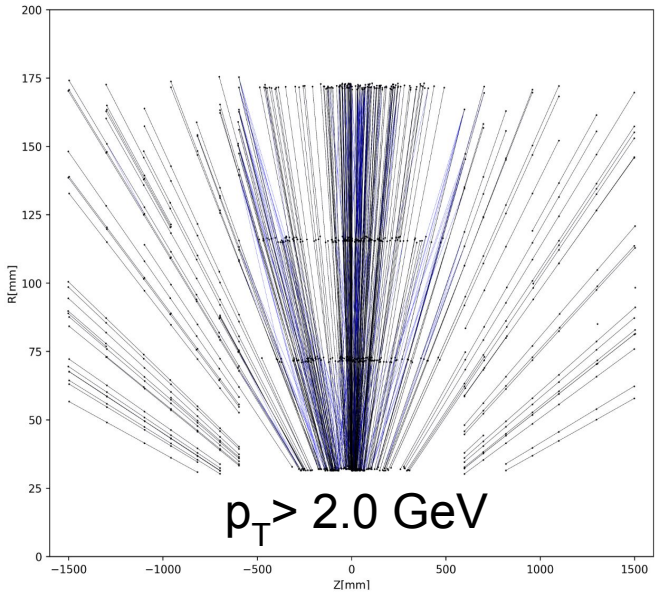


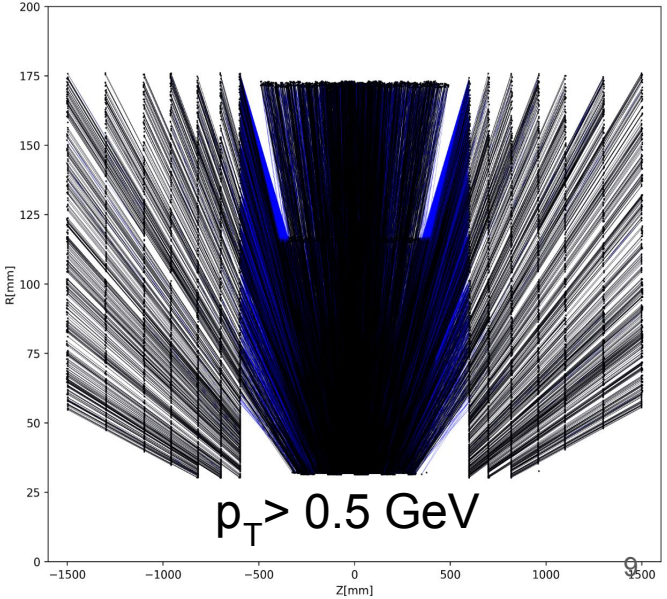
Graph Construction: Layer Pairs

- Hits are selected using truth information satisfying a p_T cut
- Edges formed between adjacent layer pairs for hit pairs with
  - $\Delta\phi/\Delta R$ < 0.0006
  - z_0 < 150 mm
  - -5 < η < 5

False edges

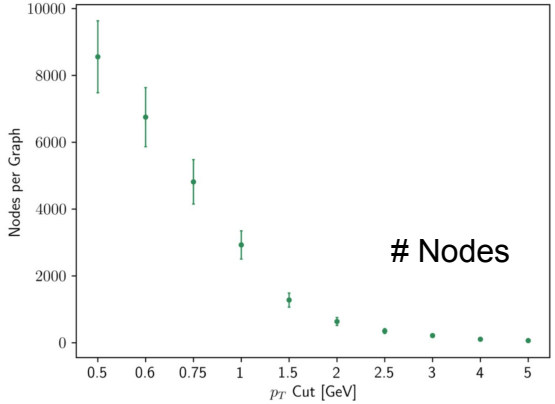
True edges

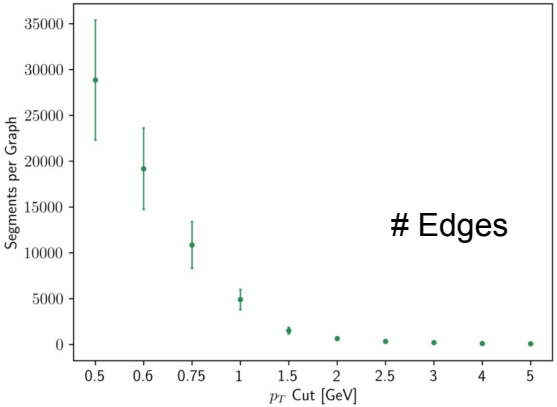




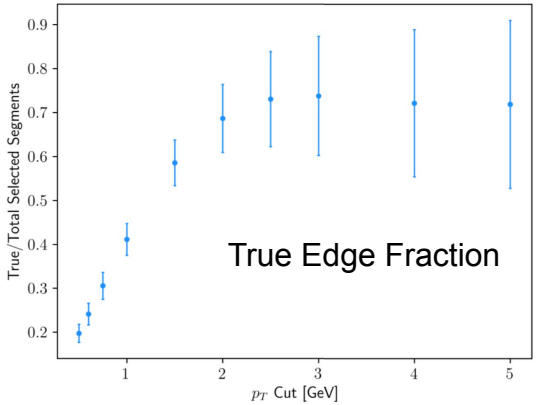
Graph Construction: Layer Pairs

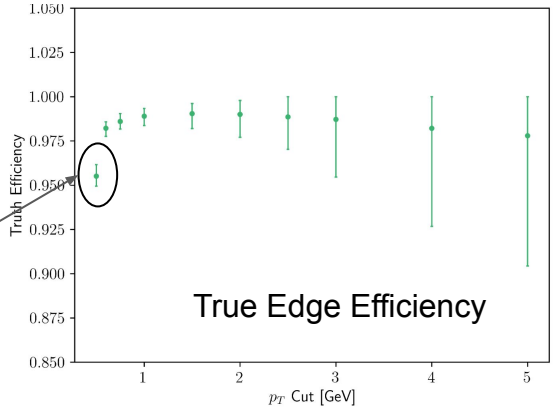
Graph Size





Graph Efficiency

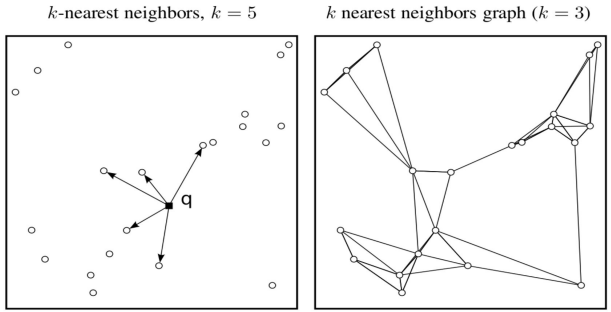


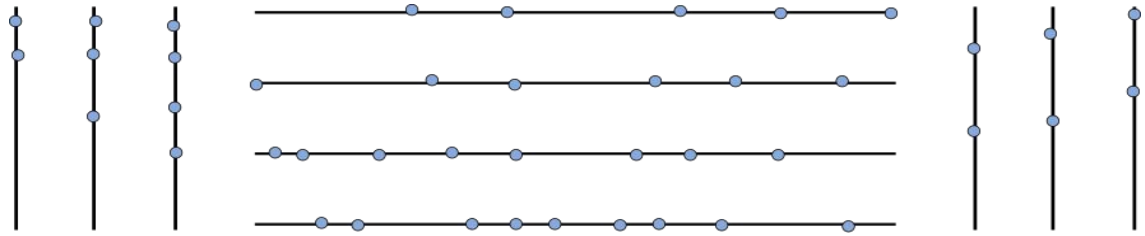


$\Delta\phi/\Delta R$ cut not optimal

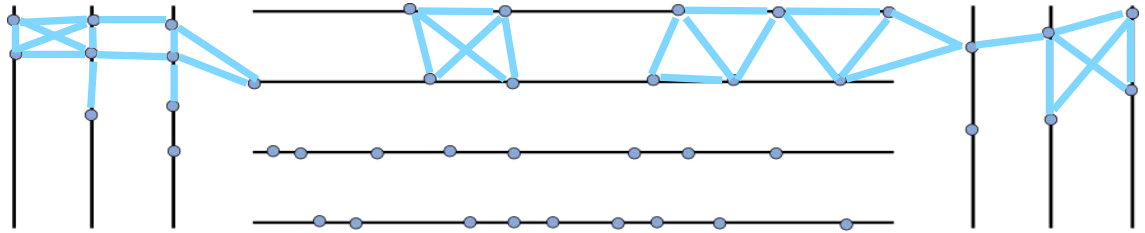
Graph Construction: kNN

- Built into pytorch geometric
- Can customize distance metric to incorporate physics information
- Implementation in progress for TrackML data set
- Need additional studies to optimize k for high pile-up





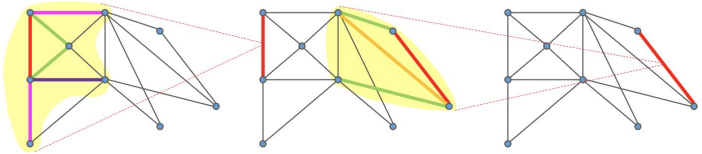
Input



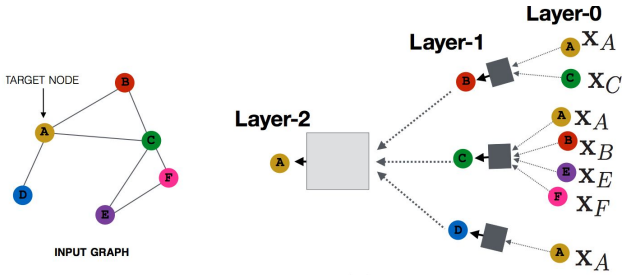
kNN with k=3 and Euclidean metric

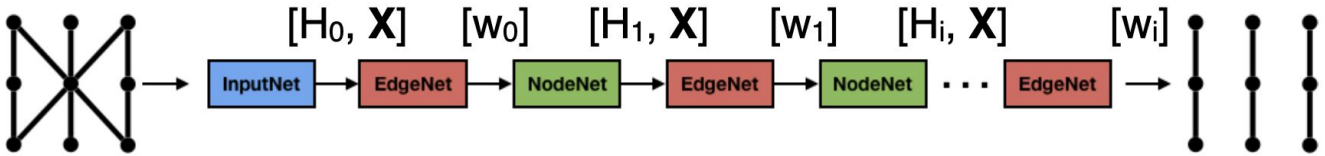
Architectures: Edge Classifier 1 (EC1)

- 'Graph Modules' combine edge and node networks
- Entire architecture is feed-forward

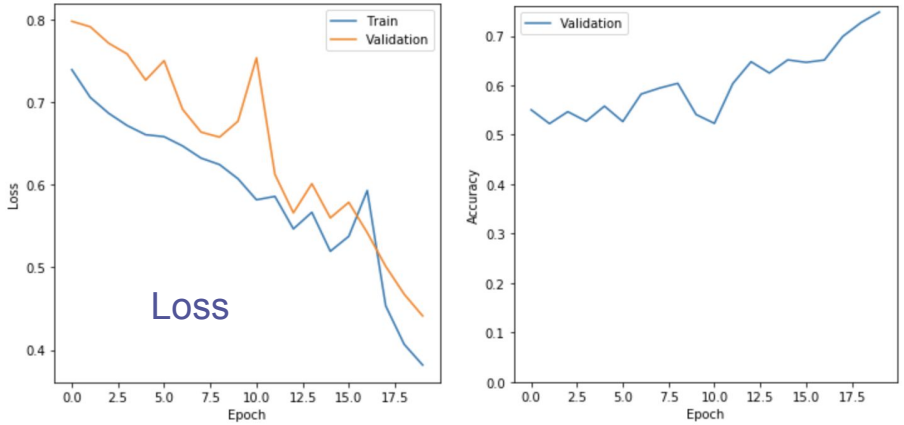


- Initial training:
  - Used layer pair graphs with 0.5 GeV p_T cut
  - Trained on Nvidia P100 GPU
  - 6 graph modules, 128 hidden dimensions,
  - Trained for 20 epochs with BCE loss and 0.001 learning rate

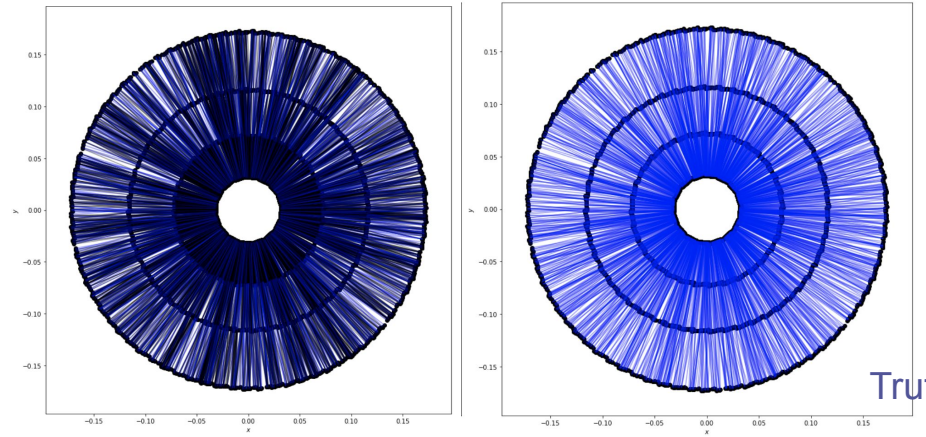




Architectures: Edge Classifier 1



Final edge accuracy: 75%
Note: network needs to be re-trained with additional epochs



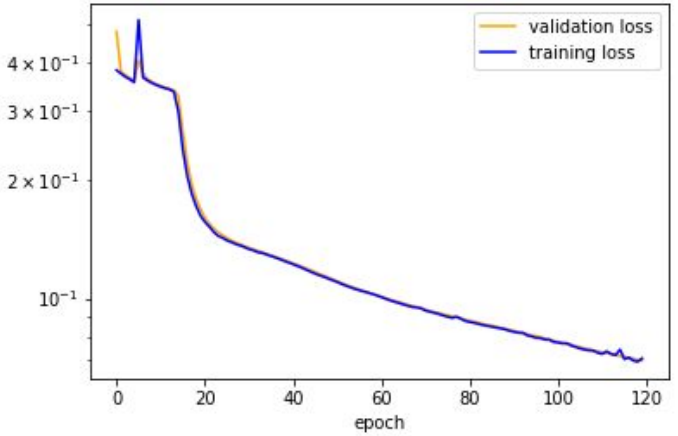
True edges: > 0.5 edge weight

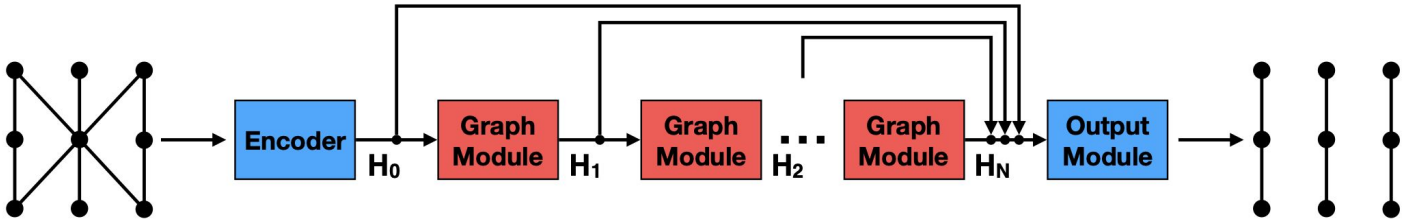
False edges: < 0.5 edge weight

Truth Graph

Architectures: Edge Classifier 2 (EC2)

- Graph embedding from each module is propagated to future modules
- Initial training:
  - Used layer pair graphs with 0.5 and 2 GeV p_T cuts
  - Trained on Nvidia GeForce 2080Ti Turbo 11G GPU
  - 6 graph modules with 64 hidden dimensions
  - Trained for 120 epochs with BCE Loss and 0.0001 learning rate

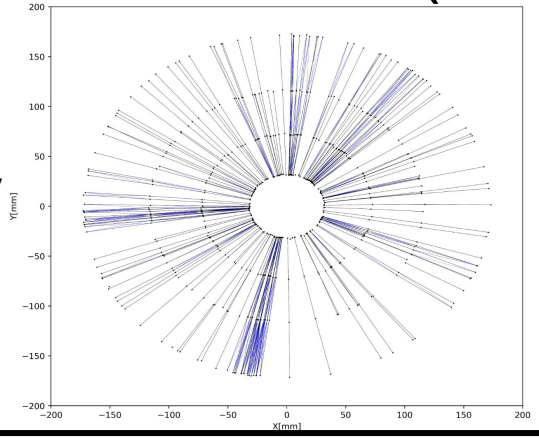




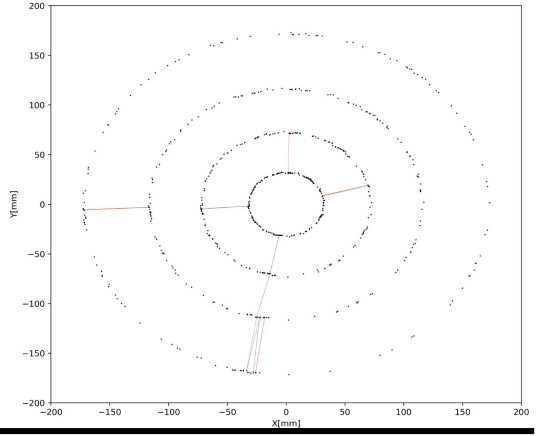
Pixel Barrel Results (EC2)

p_T > 2.0 GeV

.9824	.0120
.0176	**.9880**

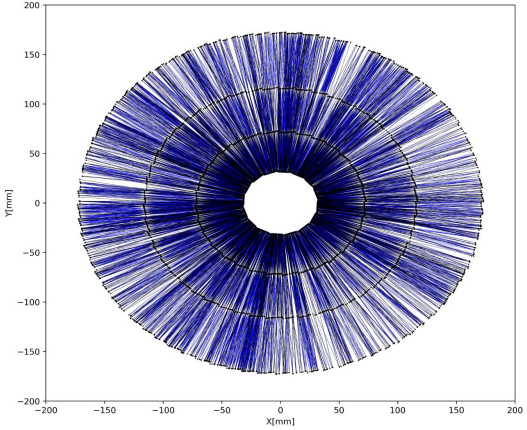


Classified Graphs:
True Edges
False Edges

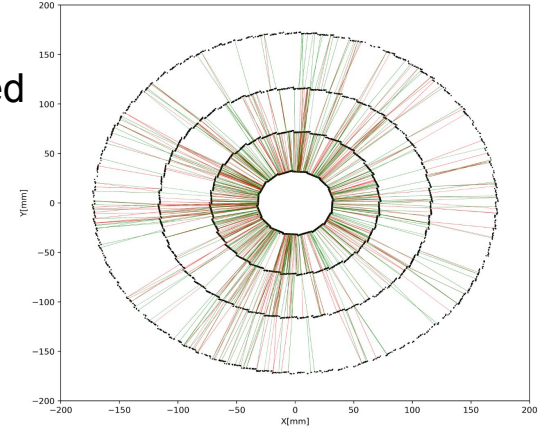


p_T > 0.5 GeV

.9790	.1004
.0210	**.8996**

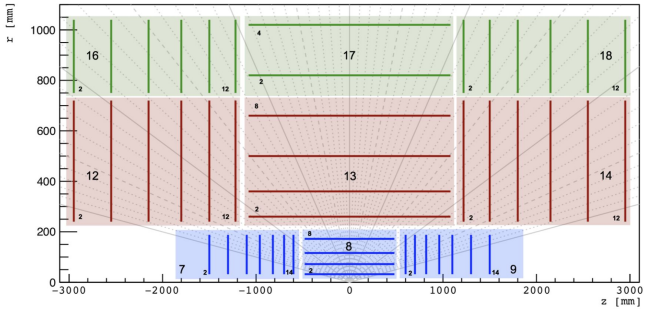


Error Graphs:
Fake Edges (classified as true, but not)
Missed Edges (truth edges classified as false)



Confusion Matrices (EC2)

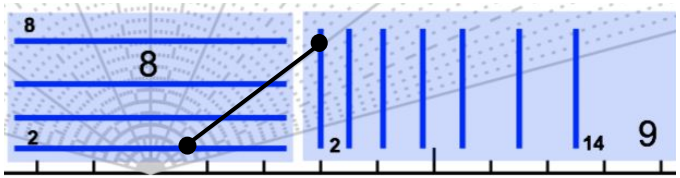
False Edge Efficiency	Missing Edge Fraction
Fake Edge Fraction	**True Edge Efficiency**

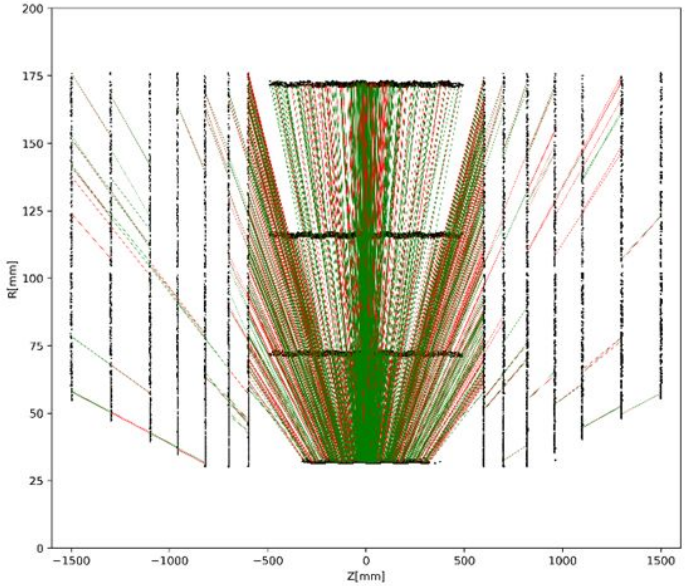


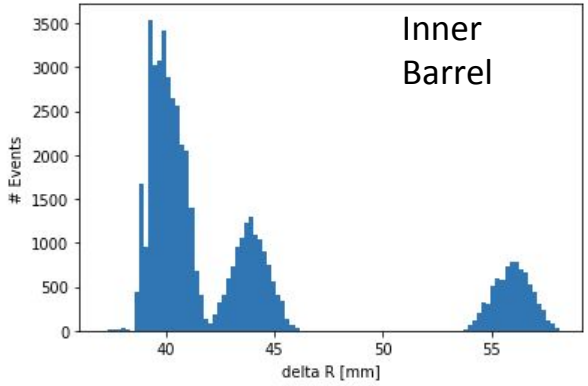
	Full Barrel	Modules 8,13,17		Inner Barrel	Module 8		Inner End Caps	Modules 7,9		Inner Detector	Modules 7,8,9
High p_T 2.0 GeV	**.9792**	.0099		**.9824**	.0120		**.9435**	.0005		**.8498**	.1050
	.0208	**.9901**		.0176	**.9880**		.0565	**.9995**		.1502	**.8950**
Low p_T 0.5 GeV	**.9840**	.0902		**.9790**	.1004		**.9882**	.0005		**.9571**	.2062
	.0160	**.9098**		.0210	**.8996**		.0118	**.9995**		.0429	**.7938**

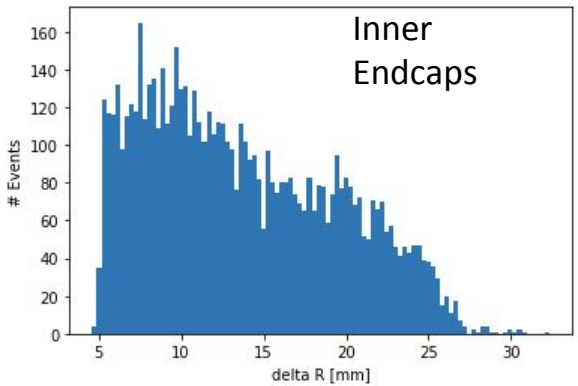
Motivation for ΔR cut (EC2)

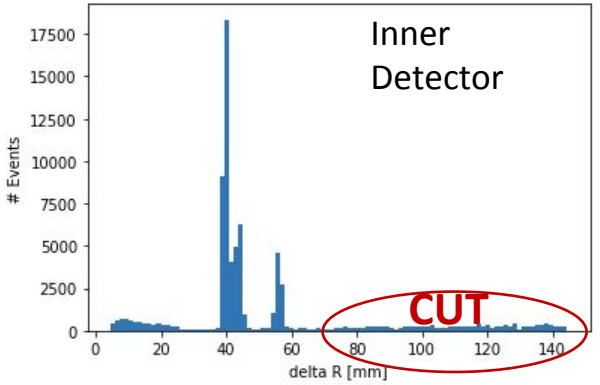
- Want to avoid edges that intersect barrel layers between them



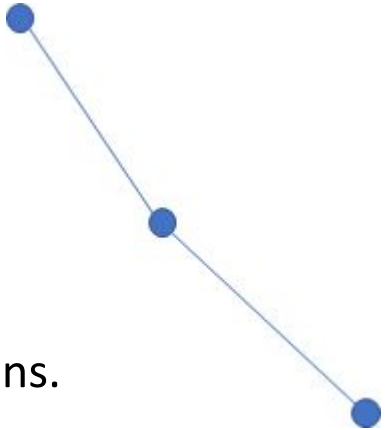








Track Building



- Initial implementation Uses UnionFind
  - Nodes are grouped into subsets by connected edges called unions.
  - Tracks are defined as any union with ≥ 3 nodes (2 edges)
- Tracks built using the truth graph (from preprocessing) and the GNN output graph (after inference)
  - GNN tracks are then matched to truth tracks.
  - Currently requires completely identical unions of nodes to define a match

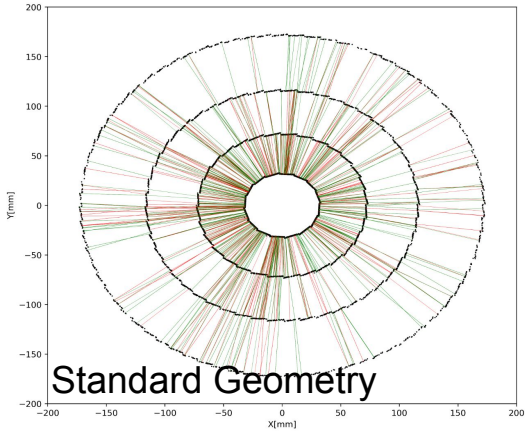
Tracking Results (EC2)

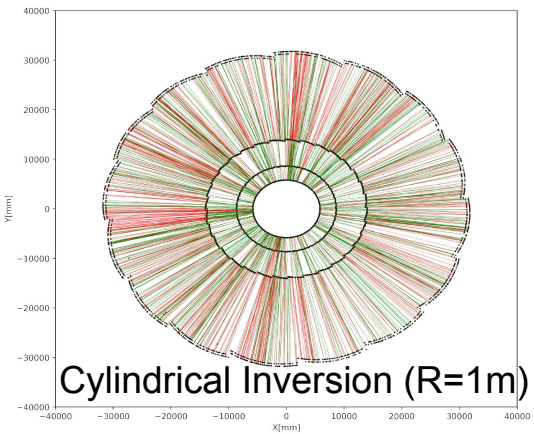
- Track Efficiency = Matched GNN Tracks / Total Truth Tracks
- Fake Fraction = Unmatched GNN Tracks / Total GNN Tracks

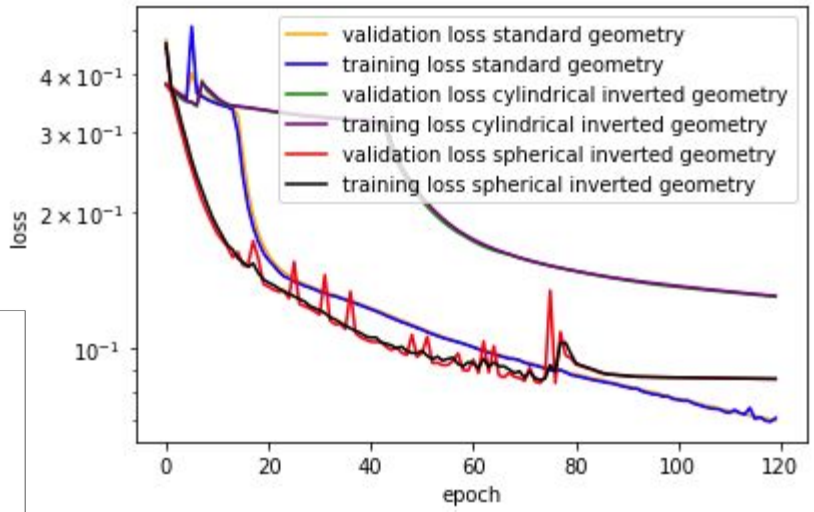
Modules Included	Full Barrel 8, 13, 17	Inner Barrel 8	Inner Endcaps 7, 9	Inner Detector 7, 8, 9	Inner Detector with dR > 65 mm cut
High p_T 2.0 GeV					
Track Efficiency	.9319	.9554	**.9975**	.6635	.6651
Fake Fraction	.0506	.0304	**.0018**	.2350	.2350
Low p_T 0.5 GeV					
Track Efficiency	.5569	.5553	**.9973**	.4788	.6288
Fake Fraction	.3678	.3494	**.0022**	.3848	.2881

Coordinate Transforms (EC2)

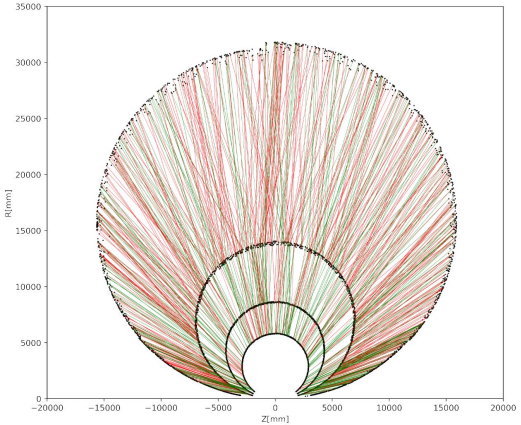
Is there a preferred geometry that would help GNN train in barrel region?





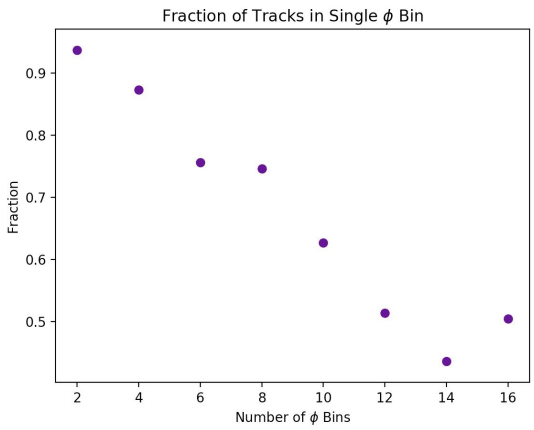


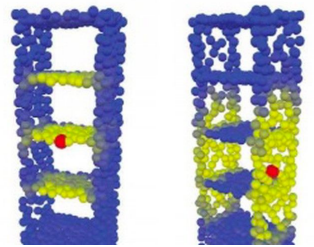
Low p_T 0.5 GeV	Standard	Cylindrical Inversion	Spherical Inversion
Track Efficiency	**.5553**	.2809	.4724
Fake Fraction	**.3494**	.5931	.4319

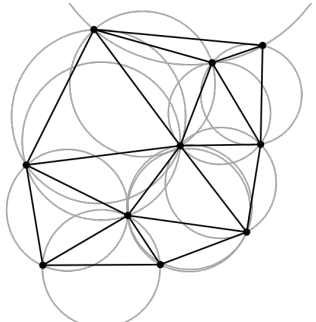


Graph Construction: On-going Work

- Optimize parameters of existing construction algorithms
  - $\Delta\phi/\Delta R$ between layers,$\Delta\phi/\Delta R$ within layers, k of kNN, etc
- Measure efficiency of graph construction-network pairs
- Assess need to segment detector for fast inference
  - Would require additional postprocessing
- Explore additional construction algorithms:
  - Dynamic knn graphs
  - Simulated annealing
  - Modified triangulation
  - Dynamic point clouds

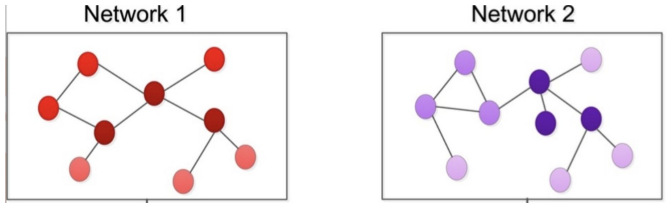


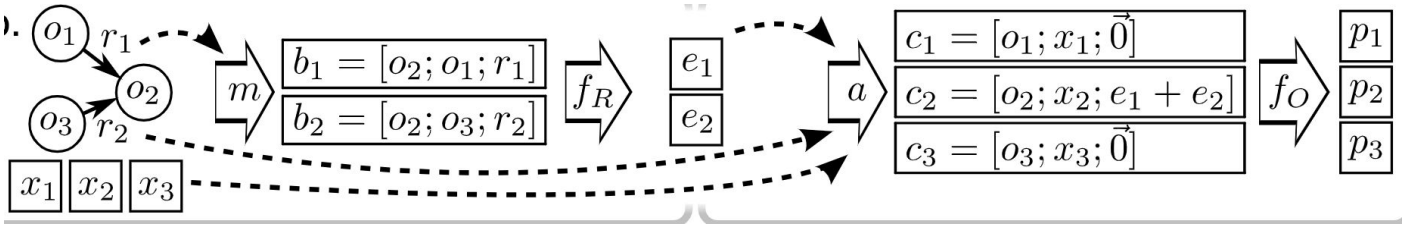




Architectures: On-going Work

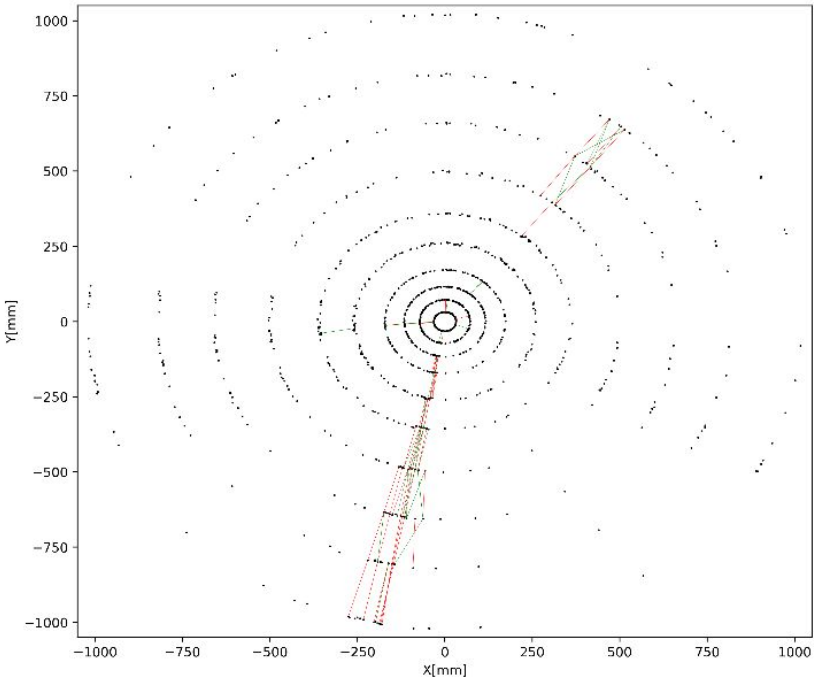
- Measure full track finding efficiency of different architectures
  - Better understand what types of edges are mis-identified by network
- Compare to existing tracking efficiencies
- Continue implementation of interaction network
  - Learns on ‘interaction relations’: two nodes and shared connecting edge
  - Users deeper NNs within each graph module, but fewer modules overall
- Explore additional architectures:
  - GraphSAGE and PinSAGE
  - Spectral Convolutions



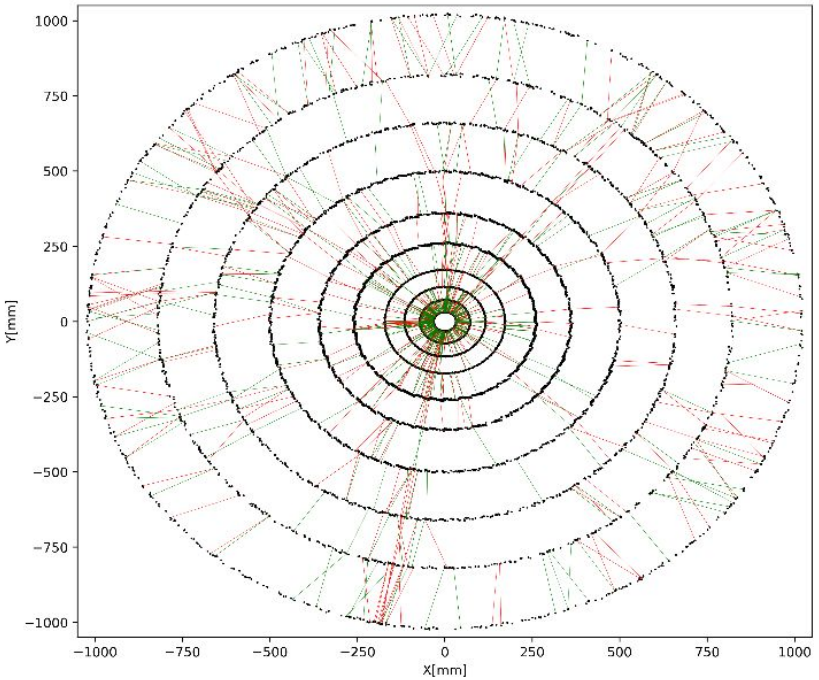


Event #8849

Full Barrel Missed Edges and Fake Edges



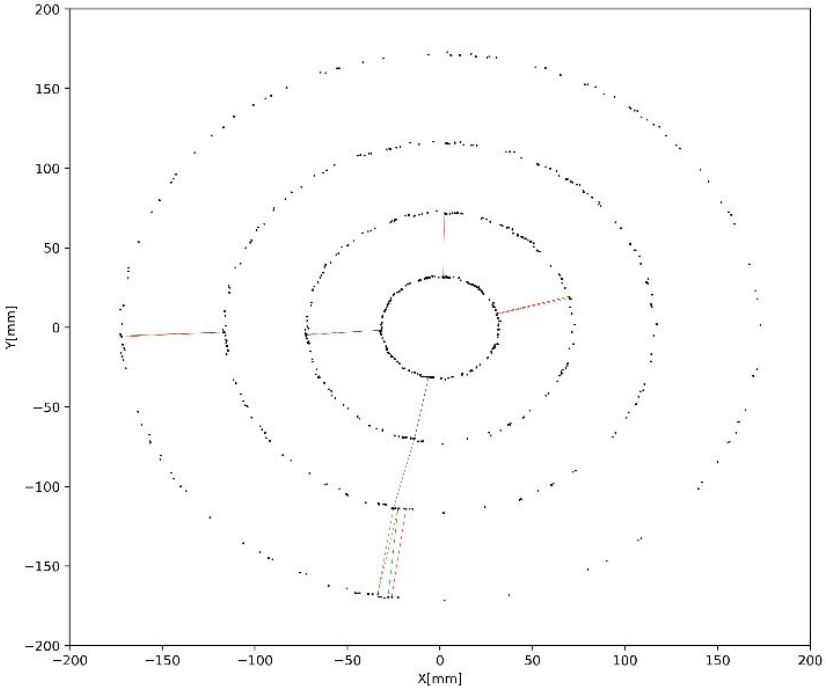
High p_T 2.0 GeV



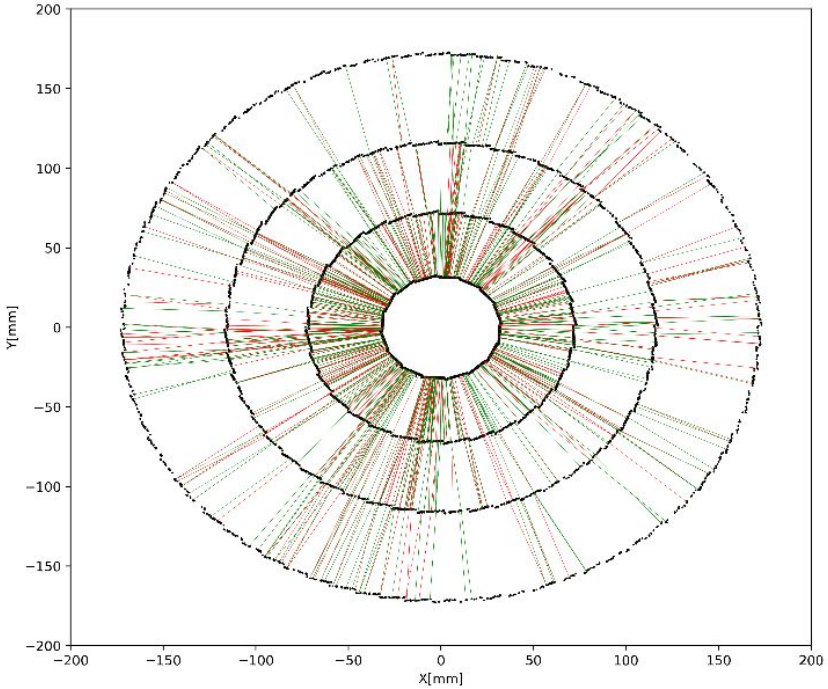
Low p_T 0.5 GeV

Event #8849

Inner Barrel Missed Edges and Fake Edges



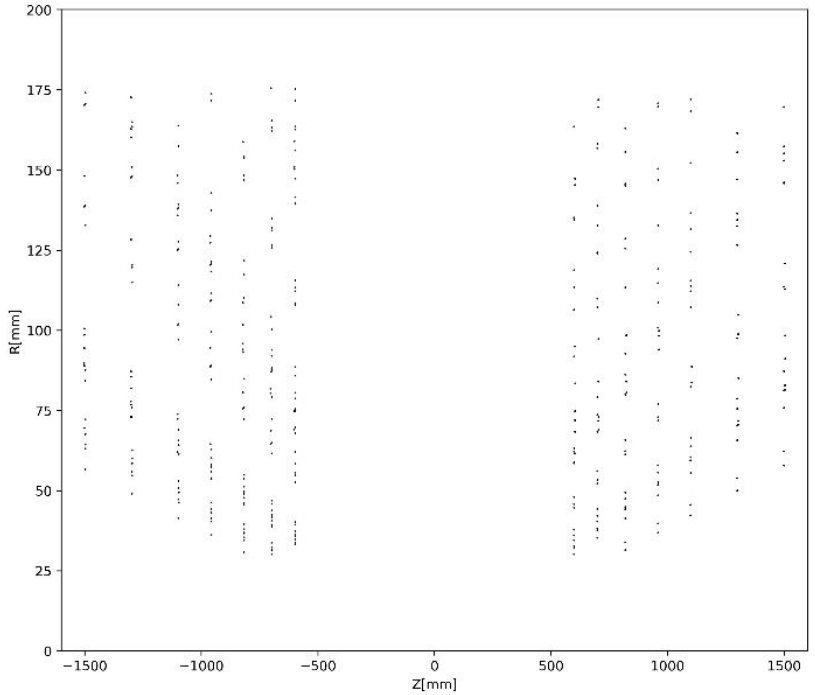
High p_T 2.0 GeV



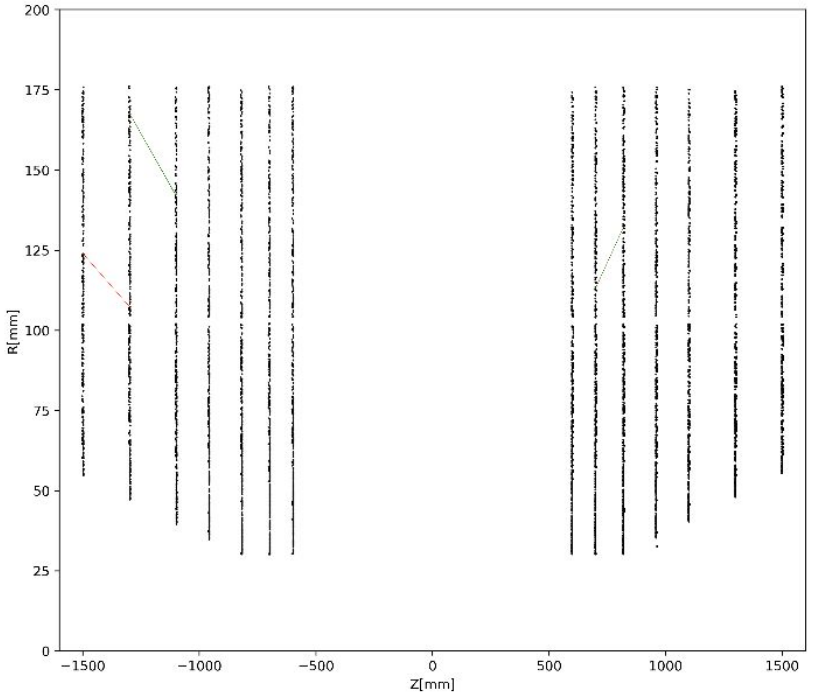
Low p_T 0.5 GeV

Event #8849

Inner Endcaps Missed Edges and Fake Edges



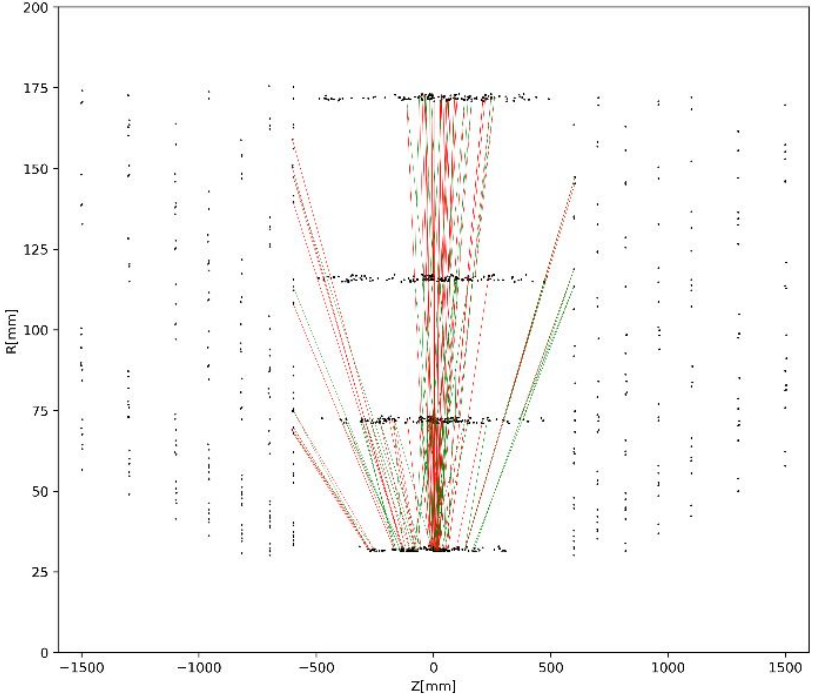
High p_T 2.0 GeV



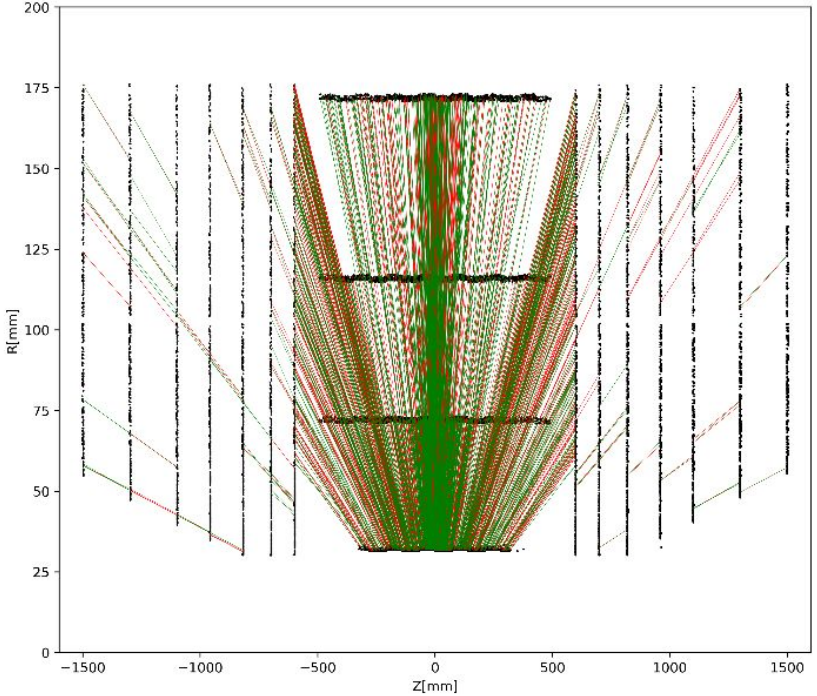
Low p_T 0.5 GeV

Event #8849

Inner Detector Missed Edges and Fake Edges



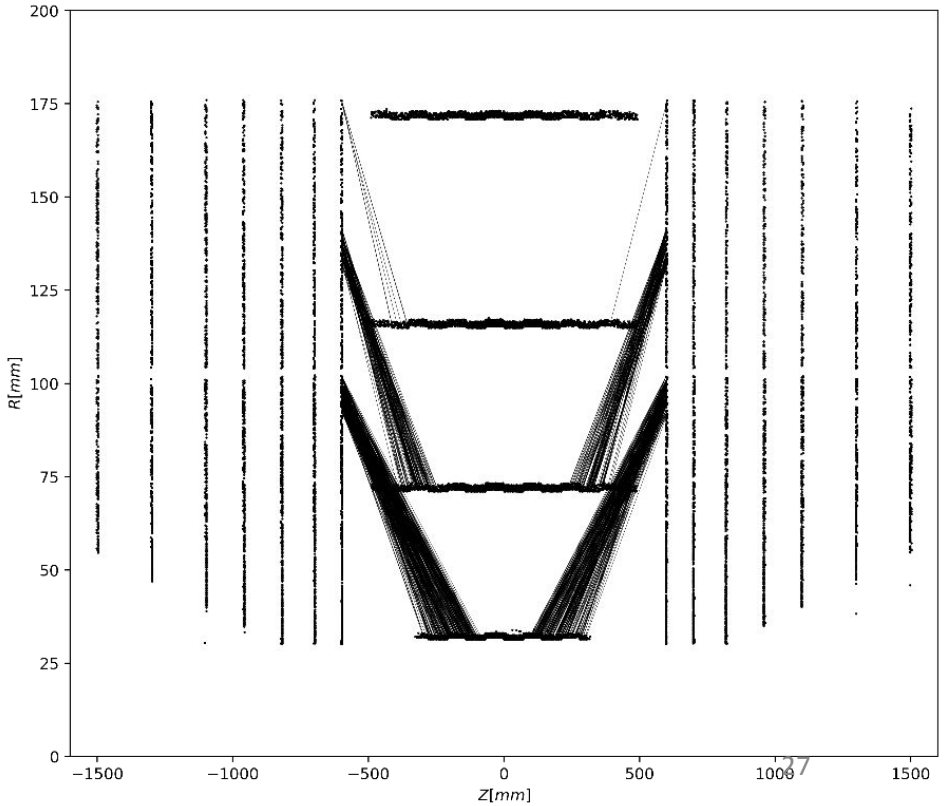
High p_T 2.0 GeV



Low p_T 0.5 GeV

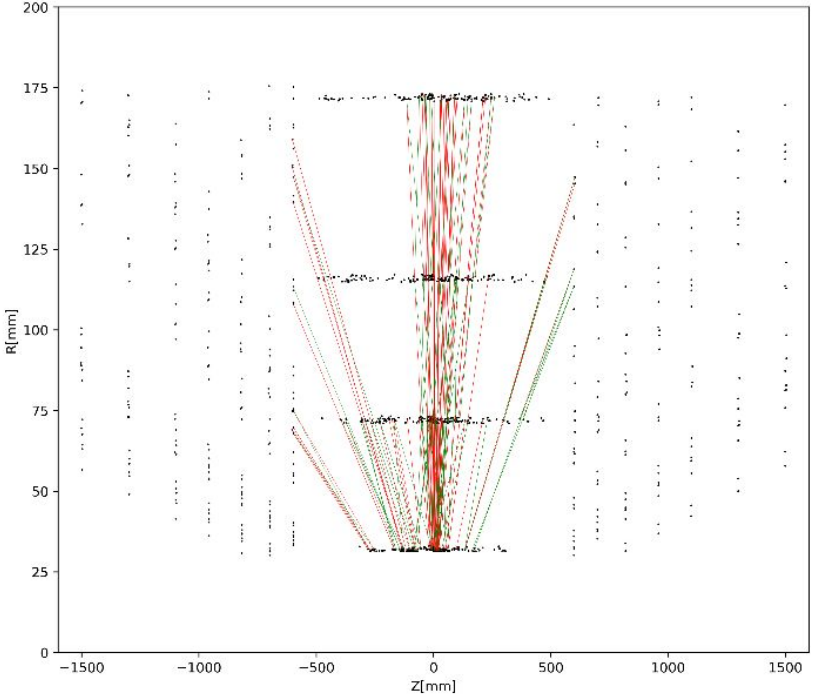
Tuning the dR cut

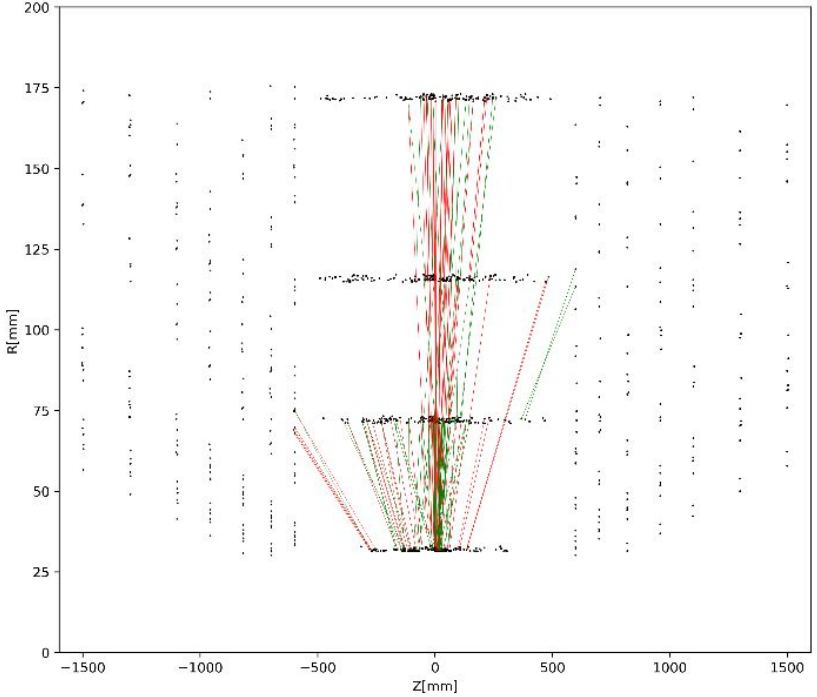
- These are all the edges that have 60mm < dR < 70mm
  - Perform a sweep in this range to find optimal value (not done)
- By eye I observed that cutting below 65mm started removing edges that we actually want to keep, but still left some of these bad edges that intersect the second barrel layer
  - Initial Cut was done at dR > 65mm



Event #8849

Inner Detector Missed Edges and Fake Edges





.8498	**.1050**
.1502	.8950

Without dR cut

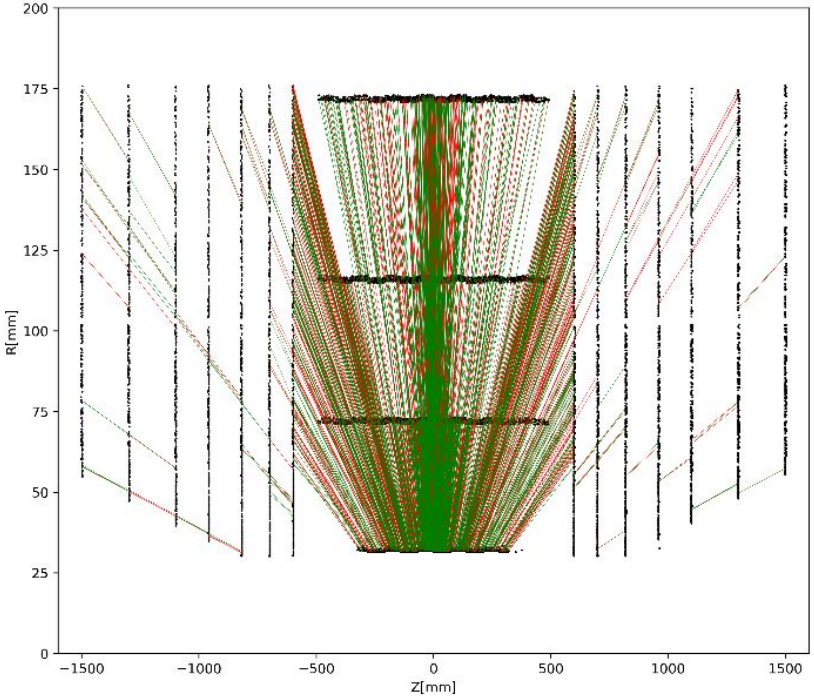
High p_T > 2.0 GeV

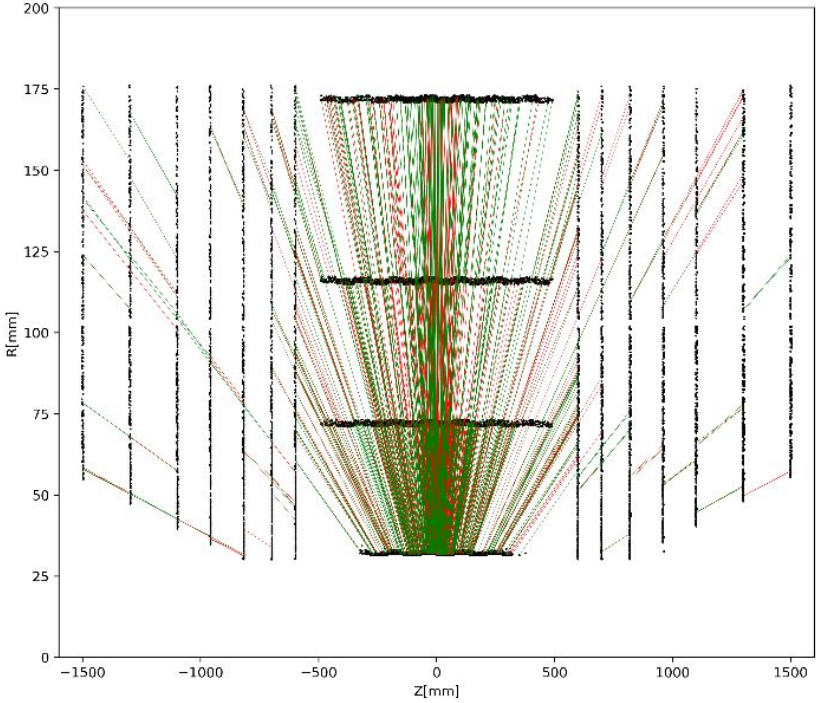
With dR > 65 mm cut

.8236	.0999
.1764	**.9001**

Event #8849

Inner Detector Missed Edges and Fake Edges





.9571	.2062
.0429	**.7938**

Without dR cut

Low p_T > 0.5 GeV

With dR > 65 mm cut

.9781	.0812
.0219	**.9188**